Rutgers Filtering Work at TREC 2002: Adaptive and Batch

Andrei Anghelescu[d], Endre Boros[i], David Lewis[m], Vladimir Menkov[a], David Neu[c] and Paul Kantor[s]
angheles@cs.rutgers.edu, boros@rutcor.rutgers.edu, ddlewis@worldnet.att.net
vmenkov@aplab.rutgers.edu, djneu@acm.org, kantor@scils.rutgers.edu *

ABSTRACT

This year at TREC 2002 we participated in the adaptive filtering sub-task of the filtering track with some models for training a Rocchio classifier. Results were poorer than average on the utility type measures. Using simple feature selection produced better than average results on an F-type measure. The key to our approach was the use of pseudo-judgments, and an approach to threshold updating. We also participated in the batch filtering sub-task of the filtering track and investigated the use of rank based feature selection techniques in conjunction with a very simple classification rule.

1. INTRODUCTION

In the adaptive filtering sub-task of the filtering track, systems utilize a training set consisting of a small set of documents which are labelled either *relevant* or *irrelevant* . This is supplemented by a training set, from which one may draw inferences about the corpus, and may hazard some conjectures as to the relevant documents. In the work reported here, a simple version of "pseudo-relevance feedback" is used to expand the terms appearing in the 3 relevant documents, and the original topic statement.

Our approach in preparing to study the problem of adaptive filtering attempts to:

–Incorporate major techniques common to high-scoring AF approaches in recent TRECs.

–Allow easy modification of aspects we are likely to be doing experiments on.

–Be efficient enough to do many tuning runs.

–Be as simple as possible to implement given the above constraints.

We prepared the AP 1988-1990 data, which served as a "sandbox" for the selection of parameters in the adaptive Rocchio model. We used the LEMUR [10] toolkit to manage the text, build indices, etc.

We introduced a number of parameters controlling how many examples will be pseudo-labeled and with what weights:

In the batch filtering sub-task of the filtering track, systems utilize a training set consisting of documents which are labelled either *relevant* or *irrelevant* for a given information needs to develop static classifiers which attempt to distinguish the documents labelled relevant from those labelled irrelevant. In our opinion, efforts to attack this problem are often complicated by several characteristics of textual data.

Textual data is generally represented by using the terms in the text as features. Such data is inherently *highly dimensional* — the number of features being potentially equal to the number of words in the English language. In addition, misspellings, the improper, or colloquial use of words, and the fact that many very common words (e.g. "a", "and", "the", etc.) are virtually useless for distinguishing relevant documents from irrelevant ones, regardless of the information need, lead textual data to be *noisy.* Finally, most terms, even those not considered "noise" under the previous description, are not needed to distinguish relevant documents from irrelevant documents.

The aforementioned characteristics of textual data indicate that it might be possible to represent a document collection using only a subset of the original feature set which is much smaller than the original feature set, yet possesses properties which serve to facilitate the process of distinguishing relevant documents from irrelevant documents.

The idea we pursued in the batch filtering sub-task of the filtering track was to employ a heuristic designed to generate such feature subset and to then train an extremely simple classifier on the training set represented only in terms of the selected feature set.

2. ADAPTIVE FILTERING: BUILDING A CLASSIFIER

*Research supported in Part by the National Science Foundation under Grant NumberEIA-0087022. PBK and KBN are supported in Part by Advanced Research and Development Activity (ARDA)'s Advanced Question Answering for Intelligence (AQUAINT) Program under contract number 2002-H790400-000, the HITIQA project, of SUNY Albany and Rutgers. EB is supported in part by the Office of Naval Research (Grant N00014-92-J-1375). The views expressed in this article are those of the authors, and do not necessarily represent the views of the sponsoring agency. Author Affiliations: $^d =^c$ Division of Computer Sciences, Rutgers, the State University of New Jersey; [i] Rutgers Center for Operations Research; ([m]). Independent Consultant, Chicago IL. [a]. Independent Consultant, Penticton, British Columbia Canada; [s] SCILS and DIMACS, Rutgers, the State University of New Jersey

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **2002**	2. REPORT TYPE	3. DATES COVERED **00-00-2002 to 00-00-2002**

4. TITLE AND SUBTITLE **Rutgers Filtering Work at TREC 2002: Adaptive and Batch**	5a. CONTRACT NUMBER 5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Rutgers, The State University of New Jersey,Piscataway,NJ,08854-8019**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S) 11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**8**	

2.1 Initialization

Initial training for each topic uses the training set (on which relevance status with respect to the topic is known only for three of the documents), and the topic description.

FOR EACH Topic

B1. Read topic description

B2. Read initial 3 positive training examples for this topic. Give each of these examples a weight of 1.0.

B3. Call scoring model learning algorithm (Rocchio) to produce linear model based on the topic description and the initial positive examples.

B4. Call the "pseudolabeling algorithm" to run the linear model trained in Step B3 against *all* training documents. It will return some portion of the training documents, and will have associated with them positive or negative pseudolabels, and fractional weights. Essentially, the documents that achieve a high score on vector retrieval with the initial query and 3 positive documents are taken as "relevant", those with low score are taken as "not relevant". Our algorithm actually has a number of parameters that control (a) the dividing line between "pseudo-relevant" and "pseudo-irrelevant" documents (which we refer to, together, as the "pseudo-labelled" documents) (b) the fractions of each class that are sampled into the updated Rocchio classifier and (c) the weights that each type of "pseudo" document are assigned in step B5. Eventually these weights are expressed in terms of the number of "equivalent documents" that the pseudo-labelled documents represent.

B5. Call the classifier learning algorithm, which changes the query, *a la* Rocchio, and selects a threshold that maximizes the target score, on the training set.

Numerous implementation details are not described here. Note that the idea of an "outer loop" over topics represents just one way to approach the problem, which may not be optimal for specific choices of the learning algorithm.

For further analytical work we have since modified the code to save the classifiers or the internal state of the training algorithm to persistent storage after initial training. This will potentially be useful for multiple experiments with the same starting point, as well as for comparative experiments (studying improvement of classifier over time).

2.2 Adaptive Phase of Training

In adaptive filtering, we run through the test documents in the specified order, applying classifiers, getting judgments only for documents judged relevant, and updating the classifiers.

FOR EACH Topic

FOR EACH Test Document

C1. Apply current classifier for topic to test document, computing score and determining if score is above threshold.

IF score is $\geq$ threshold

C2.1. Pass document ID, topic ID, score, and label ("relevant") to routine that writes output for evaluation

C2.2. Pass document ID and topic ID to judging routine, which will return label (Relevant vs. Nonrelevant vs. Unjudged).

ELSE

C3.1 Label = Unknown

C4. Pass current classifier, document ID, Label, and a weight of 1.0 to Learner (which for the baseline will be an object that in turn calls Rocchio and TROT).

2.3 the Rocchio Algorithm

The Rocchio algorithm [9] produces a linear model, which must then be specified with a threshold. The basic inputs to the algorithm are:

1. An initial "query" vector
2. A set of document vectors. Each vector is accompanied by a weight and a label.
3. The Rocchio weighting parameters (α, β, γ)
4. Feature weighting parameters
5. Feature selection parameters and rules

The Rocchio algorithm [9, 4] is a batch algorithm. It produces a new weight vector $\mathbf{w}$ from an existing weight vector w_1 and a set of training examples. The jth component w_j of the new weight vector is:

$$w_j = \alpha w_{1,j} + \beta \frac{\sum_{i \in C} x_{i,j}}{n_C} - \gamma \frac{\sum_{i \notin C} x_{i,j}}{n - n_C} \qquad (1)$$

where n is the number of training examples, $C = \{1 \leq i \leq n : y_i = 1\}$ is the set of positive training examples (i.e., members of the class of interest), and n_C is the number of positive training examples. The parameters α, β, and γ control the relative impact of the original weight vector, the positive examples, and the negative examples, respectively.

Typically, classifiers produced with the Rocchio algorithm are restricted to having nonnegative weights, so that instead of using the raw w from Equation (1), one uses $w^{'}$ where

$$w^{'} = \begin{cases} w & \text{if } w > 0 \\ 0 & \text{otherwise.} \end{cases}$$

This is turned into a classifier by the relatively expensive process of recomputing the threshold after each new judgment is received on a submitted document. The computation of the threshold can be somewhat accelerated with a full Rocchio model, but we have not found a way to accelerate it meaningfully when a non-linear step such as the selection of a number of "top features" is included.

2.4 Retaining only the top 30 terms in a query

To improve performance, we limited the number of terms appearing in a query.

The specific algorithm is given in pseudocode as

Algorithm 1: Query term selection

```
Require: query vector $Q$, k
for $t \in Q$ do
    if $t < 0$ then
        $t = 0$
    end if
end for
$S = reverse(sort(Q))$
Ensure: $S[1 : min(|S|, k)]$, the top $k$ positive components of $Q$
```

3. ADAPTIVE TRAINING HEURISTICS

To find a Rocchio classifier we started at "plausible" values for all of the parameters in the model, and conducted a "greedy" search on each of the parameter values separately.

Original results concentrated on the utility based measures, and were terrible. This led to the development of a "TREC-specific" feature, which stops sending examples for judgment if the rate of success falls too low. One such heuristic is to stop when the number of consecutive negatives exceeds the total accumulated positive judgments obtained. Such heuristics have no meaning in the real world situations to which adaptive filtering will be applied.

An alternative heuristic, which can be justified for real applications, is to reduce the number of components in the updated Rocchio vector to a very small number. In one such run the number of components is reduced to 30. These 30 components are selected on the basis of their individual explanatory power, with regard to the specific measure of performance under considerations. In the submitted run, this was an F-measure.

Since F measures can be rewritten as $\frac{1}{\beta\frac{1}{p}+(1-\beta)\frac{1}{R}}$ they are very sensitive to finding *any* relevant documents. If (g, G) are the numbers of relevant documents (found, in the collection) respectively, and n documents are returned,

$$F = 1/(\beta n/g + (1-\beta)G/g) = g/(\beta n + (1-\beta)G)$$

.

So a system that "hangs in there" and eventually produces even a single relevant document will score better than a more discriminating system that returns no relevant documents, and quits sooner.

The results of our early experiments show only that we have set up a workable laboratory for exploring a host of possible combinations of the five key ingredients of an adaptive algorithm: these ingredients are a compression rule; a representation rule; a matching scheme, a learning scheme, and a fusion or selection scheme for combining multiple approaches to each of these five components. As is well known in the information retrieval community, the adaptive filtering task is extremely difficult, but we are optimistic that previously unexplored combinations of approaches may yield meaningful improvements in performance. The results are showin in Table 2, which appears at the end of the paper. The meaning of the row and column labels is as follows.

1. label of the run, which is composed of 3 parts - the value of the weight of the unjudged documents (parameter thres.unjWt - U-xx → thres.unjWt=xx) followed by the name of the parameter that is changed and the utility that is optimised (for example "best.f" means that the f-beta utility is optimised)

   The parameter related labels have the following meanings:

   - A+ : $\alpha = 2.0$
   - A- : $\alpha = 0.5$
   - C+ : $\gamma = 0.25$
   - C- : $gamma = 0.0625$
   - ND+ : neg density s.t. 2000 pseudo negatives are selected
   - ND- : neg density s.t. 500 pseudo-negatives are selected
   - PD- : pos density = 0.5, corresponding to 10 pseudo-positive documents
   - PW+ : pos weight of 5
   - PW- : pos weight of 1
   - NW+ : neg weight of 10
   - NW- : neg weight of 2
   - def : default values, $\alpha = 1.0, \beta = 1.0, \gamma = \frac{1}{8}$, ND s.t. 1000 pseudo-negative documents are selected, PD s.t. 20 pseudo-positive documents are selected, PW = 2, NW=5%

2. the number of topics that obtained a positive score in the test
3. min score - the lowest topic score
4. the total score (sum of all topic scores)
5. average T11U score
6. average T11F score
7. average T11SU score
8. the number of topics in this run that found at least 1 positive doc
9. the number of topics in this run that found at least 3 positive docs
10. the number of topics for which at least one document was sent to the
11. the "giveup threshold" for which these results were obtained oracle.

3.1 Ratio Based Scoring

In order to provide variety, we also used an alternate scoring scheming in which documents are ordered by a measure of the ratio of their similarities to the centroids of the positive and negative examples. Thus it builds on the relevance feedback information available to Rocchio, with a key difference. Scores are calculated using the (regularized) ratio of distances between normalized vectors. Specifically, if $\mathbf{p}, \mathbf{n}$ are the unit vectors corresponding to the centroids of the positive and negative examples, and $\mathbf{d}$ is the unit vector corresponding to the document being scored, then

$$s_{\mathbf{C}}(d) = \frac{1 - (\mathbf{n}, \mathbf{d})}{1 - (\mathbf{p}, \mathbf{d})} \qquad (2)$$

If the denominator vanishes, the value 10^6 is used as a default.

In practice this was more effective with a "Quitting" rule that cust off submission if, after the first 50 documents are submitted, we have not achieved a postive utility score.

4. BATCH FILTERING: BOOLEAN MODEL

Assume that there are $n > 0$ distinct terms in the document collection and associate an index in $V = \{1, 2, \ldots, n\}$ with each of these terms. Letting $\mathbb{B} = \{0, 1\}$, we represent each document in the collection as an n-dimensional Boolean vector $\mathbf{x} \in \mathbb{B}^V$. Each component of $\mathbf{x}$ corresponds one of the distinct terms in the document collection, with $x_i = 1$ if the i^{th} term is present in the document and $x_i = 0$ if the i^{th} term is absent from the document.

For a subset $S \subseteq V$, and vector $a \in \mathbb{B}^V$, we shall let $a[S] \in \mathbb{B}^S$ denote the projection of a onto S and for $X \subseteq \mathbb{B}^V$ we shall write $X[S]$ as the projection of X on S, that is,

$X[S] = \{a[S] \mid a \in X\}$. For a subset $S \subseteq V$ let us denote by $\chi^S \in \mathbb{B}^n$ its *characteristic vector*, i.e.

$$\chi^S_j = \begin{cases} 1 & \text{if } j \in S, \\ 0 & \text{otherwise.} \end{cases}$$

We shall refer to the set of relevant documents as T and the set of irrelevant documents as F and shall assume that $T \cap F = \varnothing$, that is, there do not exist vectors $a \in T$ and $b \in F$ such that $a = b$.

A set $S \subseteq V$ is said to be a *support set* for T and F if it has the property that $T[S] \cap F[S] = \varnothing$. That is, S is a support set if each relevant document represented in terms of the selected features subset can be distinguished from each irrelevant document represented in terms of the selected features subset.

The document model described above does not preserve information about the order in which terms appear in the document and therefore is often referred to as the *bag-of-words* representation. In addition, the Boolean nature of this representation lies in contrast to a popular representation known in the information retrieval literature as the *vector space model*, in which the components x_i correspond to the (relative) frequency of the term in the document.

4.1 Measure of Separation

For a subset $S \subseteq V$, we measure the distance between the projections $T[S]$ and $F[S]$ of the sets T and $F \in \mathbb{B}^V$ onto $\mathbb{B}^S$, by the so called *average Hamming distance*. The use of Hamming distance based separation, rather than measures based on the l_1, l_2 or l_∞ norms, as is often the practice when the employing the vector space model, is suggested by the Boolean nature of our document model.

The *Hamming distance* between the vectors $a[S] \in T[S]$ and $b[S] \in F[S]$ is defined as $d_S(a,b) = \sum_{j \in S : a_j \neq b_j} 1$. The average Hamming distance between the sets $T[S]$ and $F[S]$ then is defined as

$$\Delta_{avg}(S) = \frac{1}{|T||F|} \sum_{a \in T} \sum_{b \in F} d_S(a,b). \tag{3}$$

4.2 Ranking Functions

For each $i \in V$, each of the ranking functions presented here utilizes the following four values

- $a_i \equiv$ the number of relevant documents containing the i^{th} term
- $b_i \equiv$ the number of irrelevant documents containing the i^{th} term
- $c_i \equiv$ the number of relevant documents which do not contain the i^{th} term
- $d_i \equiv$ the number of irrelevant documents which do not contain the i^{th} term

For each $i \in V$, the relationship between a_i b_i, c_i and d_i and the document collection is given by the following 2×2 contingency table

	$y \in T$	$y \in F$	
$x_i = 1$	a_i	b_i	$a_i + b_i = \theta_i$
$x_i = 0$	c_i	d_i	$c_i + d_i = \overline{\theta}_i$
	$a_i + c_i = \|T\|$	$b_i + d_i = \|F\|$	m

where the marginals θ and $\overline{\theta}_i$ represent the number of documents containing the i^{th} term and the number of documents which do not contain the i^{th} term respectively, and $y \in \mathbb{B}$ is defined as

$$y = \begin{cases} 1 & \text{if } x \in T, \\ 0 & \text{otherwise.} \end{cases}$$

Obviously, the marginals $|T|$ and $|F|$ are constant for all terms while the marginals θ_i and $\overline{\theta}_i$ vary for each term. The total number of documents in the collection is $m = a_i + b_i + c_i + d_i$ which is obviously also a constant.

For the simplicity of notations, we shall view all ranking functions as functions of the four parameters a, b, c and d, though clearly there are only two independent values among these.

In [2] we analyzed and compared a number of possible ranking functions, and based on that study, we selected 5 such functions for this TREC experiment:

Function α

$$\alpha \;\; = \;\; \left| \frac{a}{a+c} - \frac{b}{b+d} \right| = \frac{|ad - bc|}{|T||F|} \tag{4}$$

is the absolute value of the difference between the number of relevant-irrelevant document pairs in the training collection which provide evidence that the i^{th} term is a good classifier of relevant documents and the number of relevant-irrelevant document pairs which provide evidence that the i^{th} term is a good classifier of irrelevant documents, normalized by the total number (i.e. both correctly distinguished and incorrectly distinguished) of relevant-irrelevant document pairs.

Function β

$$\beta = \frac{ad + bc}{(a+c)(b+d)} = \frac{ad + bc}{ab + ad + bc + cd} = \frac{ad + bc}{|T||F|} \tag{5}$$

is the total number of relevant–irrelevant document pairs correctly distinguished by the i^{th} term, normalized by the total number of relevant-irrelevant document pairs in the training collection.

Function γ

$$\gamma = \frac{ad}{(a+c)(b+d)} = \frac{ad}{|T||F|}$$

is an obvious variant of both α and *beta*.

Function δ

$$\delta = \frac{|ad - bc|}{\sqrt{(a+b)(c+d)(a+c)(b+d)}} = \frac{ad - bc}{\sqrt{\theta\overline{\theta}|T||F|}} \tag{6}$$

is the absolute value of the *Pearson Product Moment Correlation* coefficient or simply the *correlation coefficient* for the Boolean variables x_i and y as defined above. It measures the degree to which these two variables have a linear relationship.

Function ρ

$$\rho = \frac{(a+b+c+d)\;(ad - bc)^2}{(a+b)(c+d)(a+c)(b+d)} = \frac{m(ad - bc)^2}{\theta\overline{\theta}|T||F|} \tag{7}$$

is the χ^2 statistic for the Boolean variables x_i and y as defined above and provides another measure of association for these two variables.

Note that ρ is a onotone funcion of δ, so that our precedure, as described below, effectively gives a "double weight" to this particular measure of effectiveness.

4.3 Training the Batch Classifier

This section describes the feature selection method and the simple classifier used in the batch filtering sub-task of the filtering track.

The set of unique terms in the training set $T \cup F$ was ranked by each of the five ranking functions $\alpha, \beta, \gamma, \delta, \rho$ described in §4.2. Five intermediate feature sets, S_α, S_β, S_γ, S_δ, S_ρ, were constructed using the top ranking $K = 50$ terms of the corresponding ranking functions. Letting

$$\tilde{S} = S_\alpha \cup S_\beta \cup S_\gamma \cup S_\delta \cup S_\rho$$

we assigned a score $\psi \in \{1, \cdots, 5\}$ to each of the terms in $tiledS$, defined as the number of sets S_ξ, $\xi \in \{\alpha, \beta, \gamma, \delta, \rho\}$ in which the term appeared. The final feature set S was constructed by selecting the $K = 50$ terms with the highest ψ scores.

Next, to each term in $i \in S$ we assigned the weight

$$\omega(i) = \frac{\frac{a_i+0.5}{a_i+c_i+1}}{\frac{b_i+0.5}{b_i+d_i+1}}$$

which can be seen to be the Bayesian weight of evidence, and to each document $y \in T[S] \cup F[S]$ we assigned the score

$$\Omega(y) = \sum_{j \in S} \log(\omega(i)) y_j.$$

That is, each document projected onto the selected feature set S is assigned a score equal to the sum of the logarithms of the Bayesian weights of evidence for the terms it contains.

The batch filtering task requires the definition of a static classification rule which specifies whether each document in the test set should be considered relevant and retrieved, or irrelevant and ignored. The rule we utilized specifies that $y \in T[S] \cup F[S]$ will be retrieved if and only if $\Omega(y) \geq \tau$ for some $\tau \in \mathbb{R}$. The threshold τ was selected so as to optimize the utility measure $TU11 = 2R - I$ over the training set, where R is the number of relevant documents retrieved by the system and I is the number of irrelevant documents retrieved by the system.

5. RESULTS

5.1 Filtering Results

Our training results, using a variety scoring measures, for a great variety of training runs, are shown at the end of the paper in Table 2. In the final analysis, our results at TREC were in the middle of the pack.

These are summarized in Table 1.

Method	Mean T11
dimacsddl02a	0.110
dimacs11aAPQ	0.142
dimacsddl02b	0.293
dimacs11aP1Q	0.272
dimacs11a30Q	0.337

Table 1: TREC 2002 Results for the Assessor topics, various runs

The best results were achieved by the run submitted as $dimacs11a30Q$. This was a Rocchio method, trained on a set of documents similar to the one used at TREC. The “ 30” indicates that only the top 30 terms, that is, the 30 terms with highest weight in the updated query vector were included. “ AP” includes only the terms with positive weight are retained. “ Q” indicates that for our final submission we cut off submission if we did not achieve a positive score after submitting 50 documents for judgment. “ P1Q” used a ratio scoring scheme, together with the “ quit at 50 if score is negative rule. This is, of course, a “ TREC strategy” and not a procedure that would be useful in a real world application.

We have subsequently learned that with proper learning parameters, as chosen by the group from the Chinese Academy of Sciences, it is possible for a Rocchio approach similar to ours to achieve very good results. We are not certain as to which steps of our approch blocked us from realizing this high level of performance. One possibility is that even the small number of pseudo-negative cases that we introduce into the training is sufficient to keep us away from the region of good performance. Another is that the space of parameters is too large, and the dependence of the learning too complex, to be successfully explored “one variable at a time”, which was essentially the heuristic used. Other inhibiting factors may have included the heursitcs used to cut off submission if we did not achieve a positive score after the first 50 judgments. Nonetheless, our submission that *did* use this heuristic fared better than those that did not.

5.2 Batch Results

On the *assessor judged topics* our TU11 score was less than the median fifteen times, equal to the median twenty times, and greater than the median fifteen times and never attained the maximum.

On the *intersection topics* our TU11 score was less than the median once, equal to the median six times, and greater than the median forty-three times and attained the the maximum twenty-one times. Unfortunately, for many of these topics, submitting no documents at all was an effective TREC strategy.

6. CONCLUSIONS

This work is part of a larger effort to develop an array of approaches to filtering problems, and to integrate or fuse them for greater effectiveness. In this first effort it would appear that we have adopted tools that are capable of “state of the art” perfromance on the adaptive filtering task, but have not yet learned how to ensure that this level of performance is achieved.

7. ACKNOWLEDGMENTS

This research is supported in part by the National Science Foundation, which is not, however, responsible for any positions expressed in this report. We thank our colleagues in the Monitoring Message Streams project: Fred Roberts (Principal Investigator), David Madigan, Ilya Muchnik, S. Muthukrishnan, Rafi Ostrovsky, and Martin Strauss for helpful conversations.

8. REFERENCES

[1] Endre Boros, Takashi Horiyama, Toshihide Ibaraki, Kazuhisa Makino, and Mutsunori Yagiura. Finding essential attributes from binary data. *Annals of Mathematics and Artificial Intelligence*, accepted.

[2] Endre Boros and David J. Neu. Rank based feature selection in information retrieval. Technical report, RUTCOR, Rutgers University, 2002.

[3] William B. Frakes and Ricardo Baeza-Yates, editors. *Information Retrieval: Data Structures and Algorithms*. Prentice-Hall PTR, 1992.

[4] Donna Harman. Relevance feedback and other query modification techniques. In William B. Frakes and Ricardo Baeza-Yates, editors, *Information Retrieval: Data Structures and Algorithms*, pages 241–263. Prentice Hall, Englewood Cliffs, NJ, 1992.

[5] D. D. Lewis. Feature Selection and Feature Extraction for Text Categorization. In *Proceedings of Speech and Natural Language Workshop*, pages 212–217, San Mateo, California, 1992. Morgan Kaufmann.

[6] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[7] Gottfried E. Noether. *Introduction to Statistics: The Nonparametric Way*. Springer-Verlag, 1991.

[8] S.E. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–146, 1976.

[9] J. J. Rocchio, Jr. Relevance feedback in information retrieval. In Gerard Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 313–323. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971.

[10] Chengxiang Zhai, Thi Nhu Truong, John Lafferty, Jamie Callan, David Fisher, Fangfang Feng, James Allan, and Bruce Croft. *Lemur*. http://www-2.cs.cmu.edu/ lemur, 2001.

Test label	2	3	4	5	6	7	8	9	10	11
U-0.0_A+	9	-83	67	1.340	0.166	0.266	27	24	50	n/a
U-0.0_A-	8	-234	-529	-10.796	0.158	0.250	31	23	49	n/a
U-0.0_C+	12	-67	109	2.180	0.195	0.274	35	30	50	n/a
U-0.0_C-	13	-86	-79	-1.580	0.175	0.268	30	23	50	n/a
U-0.0_ND+	14	-77	143	2.860	0.191	0.278	33	25	50	n/a
U-0.0_ND-	14	-77	120	2.400	0.190	0.275	34	25	50	n/a
U-0.0_NW+	13	-80	53	1.060	0.179	0.282	30	24	50	n/a
U-0.0_NW-	11	-109	-142	-2.840	0.173	0.269	31	25	50	n/a
U-0.0_PD-	11	-61	49	1.043	0.168	0.273	27	20	47	n/a
U-0.0_PW+	8	-374	-703	-14.060	0.165	0.224	33	26	50	n/a
U-0.0_def	16	-51	325	6.771	0.191	0.287	32	23	48	n/a
U-0.1_A+.best.f	15	-34	499	9.9800	0.2040	0.3030	34	26	50	184
U-0.1_A-.best.f	15	-63	103	2.1460	0.1730	0.2690	30	22	48	293
U-0.1_C+.best.f	17	-44	347	7.2290	0.2110	0.2880	36	30	48	161
U-0.1_C-.best.f	11	-54	260	5.4170	0.1750	0.2850	28	20	48	186
U-0.1_ND+.best.f	17	-50	405	8.4380	0.1940	0.2930	32	23	48	211
U-0.1_ND-.best.f	16	-50	370	7.7080	0.1930	0.2900	32	23	48	212
U-0.1_NW+.best.f	12	-50	338	7.0420	0.1870	0.2890	30	22	48	249
U-0.1_NW-.best.f	13	-50	317	6.6040	0.1900	0.2880	32	24	48	192
U-0.1_PD-.best.f	13	-56	242	5.1490	0.1630	0.2960	27	18	47	147
U-0.1_PW+.best.f	14	-65	181	3.6200	0.1960	0.2800	34	25	50	217
U-0.1_PW-.best.f	15	-30	331	6.8960	0.1820	0.2900	30	24	48	191
U-0.1_def.best.f	16	-50	368	7.6670	0.1930	0.2900	32	23	48	211
U-0.25_A+.best.f	16	-34	567	11.3400	0.2100	0.3060	34	26	50	176
U-0.25_A-.best.f	16	-63	144	3.0000	0.1780	0.2740	30	22	48	235
U-0.25_C+.best.f	16	-44	381	7.9380	0.2100	0.2910	35	30	48	157
U-0.25_C-.best.f	12	-53	278	5.7920	0.1750	0.2860	28	20	48	184
U-0.25_ND+.best.f	17	-49	443	9.2290	0.1960	0.2950	32	24	48	244
U-0.25_ND-.best.f	16	-49	395	8.2290	0.1950	0.2920	32	24	48	216
U-0.25_NW+.best.f	12	-50	353	7.3540	0.1880	0.2900	30	22	48	246
U-0.25_NW-.best.f	12	-48	313	6.5210	0.1920	0.2900	32	25	48	335
U-0.25_PD-.best.f	13	-55	248	5.2770	0.1630	0.2960	27	18	47	146
U-0.25_PW+.best.f	14	-70	141	2.8200	0.1930	0.2810	34	24	50	188
U-0.25_PW-.best.f	15	-30	356	7.4170	0.1820	0.2910	30	23	48	194
U-0.25_def.best.f	16	-49	394	8.2080	0.1940	0.2920	32	24	48	210
U-0.5_A+.best.f	16	-33	566	11.3200	0.2100	0.3070	34	25	50	176
U-0.5_A-.best.f	15	-52	185	3.8540	0.1800	0.2760	30	23	48	231
U-0.5_C+.best.f	16	-41	390	8.1250	0.2060	0.2920	34	28	48	313
U-0.5_C-.best.f	12	-35	306	6.3750	0.1730	0.2880	27	20	48	185
U-0.5_ND+.best.f	17	-48	468	9.7500	0.1950	0.2960	32	24	48	223
U-0.5_ND-.best.f	16	-48	418	8.7080	0.1940	0.2930	32	24	48	223
U-0.5_NW+.best.f	12	-49	369	7.6880	0.1870	0.2920	29	22	48	252
U-0.5_NW-.best.f	13	-47	291	6.0620	0.1890	0.2860	32	25	48	337
U-0.5_PD-.best.f	13	-54	258	5.4890	0.1640	0.2970	27	18	47	143
U-0.5_PW+.best.f	13	-68	121	2.4200	0.1910	0.2800	34	23	50	334
U-0.5_PW-.best.f	15	-27	329	6.8540	0.1820	0.2910	30	23	48	212
U-0.5_def.best.f	16	-48	411	8.5620	0.1930	0.2930	32	24	48	222
U-0.75_A+.best.f	15	-33	557	11.1400	0.2070	0.3060	34	25	50	174
U-0.75_A-.best.f	16	-61	175	3.6460	0.1850	0.2740	31	24	48	209
U-0.75_C+.best.f	17	-41	436	9.0830	0.2090	0.2990	34	28	48	319
U-0.75_C-.best.f	12	-35	358	7.4580	0.1740	0.2910	27	20	48	185

continued on the next page

continued from the previous page

Test label	2	3	4	5	6	7	8	9	10	11
U-0.75_ND+.best.f	17	-47	473	9.8540	0.1990	0.2970	32	24	48	231
U-0.75_ND-.best.f	16	-47	430	8.9580	0.1980	0.2940	32	24	48	212
U-0.75_NW+.best.f	12	-48	377	7.8540	0.1900	0.2930	29	23	48	256
U-0.75_NW-.best.f	13	-47	314	6.5420	0.1910	0.2860	32	25	48	169
U-0.75_PD-.best.f	13	-54	273	5.8090	0.1640	0.2980	27	18	47	195
U-0.75_PW+.best.f	14	-67	181	3.6200	0.1930	0.2830	34	23	50	206
U-0.75_PW-.best.f	15	-26	322	6.7080	0.1850	0.2930	30	24	48	170
U-0.75_def.best.f	16	-47	427	8.8960	0.1980	0.2940	32	24	48	212

Table 2: Utility scores of running Rocchio with different parameters